CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

CE143: COMPUTER CONCEPTS & PROGRAMMING

Chapter – 3

Operators and Expression in ‘C’

DEPSTAR
Devang Patel Institute of Advance Technology and Research

Objectives

- To learn arithmetic, string, relational, logical, bit wise, unary, binary, ternary operators
- To learn about implicit/explicit conversions between values of different types
- To learn how to build complicated expressions from operators and functions
- To learn how operator precedence/associativity control order of evaluation

Introduction

- In this chapter, we will discuss
  - Classification of Operators
  - Arithmetic Expression & Evaluation
  - Type Conversion : Implicit & Explicit
  - Associativity & Precedence

Operators

- **Operator** is a symbol that tells the computer to perform certain mathematical or logical manipulations.
- Operators are used in programs to manipulate the data and variables.
- They usually form a part of the mathematical or logical expressions.
- Some operator required 2 operands (Binary Operator) to perform operation or Some required single operand (Unary) to perform operation.

C operators are classified into no. of categories

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Increment/Decrement Operators
6. Bitwise Operators
7. Conditional Operators
8. Special Operators

Classification of Operator Continued…

Arithmetic Operator

- This operator used for numeric calculation.
- There are 2 types of arithmetic operators in C:

1. unary operators : operators that require only one operand.

2. binary operators : operators that require two operands.

Classification of Operator Continued…

Unary Operator

C Operation	Operator	Example
Positive	+	a = +3
Negative	-	b = -a
Increment	++	i++
Decrement	--	i--

- The first assigns positive 3 to a
- The second assigns the negative value of a to b.
- i++ is equivalent to i= i+ 1
- i--is equivalent to i= i-1

Classification of Operator Continued…

Binary Operator

C Operation	Operator	Example
Addition	+	a + b
Subtraction	-	a - b
Division	/	a / b
Multiplication	*	a * b
Modulus	%	a % b

- The division of variables of type int will always produce a variable of type int as the result.
- You could **only use** modulus (%) operation on int variables.

- **Arithmetic operators are further classified as:**
  - Integer Arithmetic
    - 10/5 = 2
  - Real Arithmetic
    - 10.2 + 8.9 = 19.1
  - Mixed Mode
    - 10.5/2 = 5.25

Classification of Operator Continued…

Relational Operator

- It is use to compared value of two expressions depending on their relation.

Associativity = Left to right

Operator	Example	Meaning
>	x > y	x is greater than y
<	x < y	x is less than y
>=	x>=y	x is greater than or equal to y
<=	x<=y	x is less than or equal to y
==	x == y	equal to
!=	x != y	not equal to

Classification of Operator Continued…

Logical Operator

- Operator used with one or more operand and return either value zero (for false) or one (for true).
- The operand may be constant, variables or expressions.

Associativity examples:

https://www.programiz.com/c-programming/precedence-associativity-operators

Operator	Example	Meaning
&&	a<b && a<c	AND
||	a>b || a>c	OR
!	!(a<c)	NOT

- Where logical NOT is a unary operator and other two are binary operator. Logical **AND** gives result **true** if **both** the conditions are **true**, otherwise result is false. And logical **OR** gives result **false** if both the condition **false**, **otherwise** result is **true**.

Classification of Operator Continued...

Assignment Operator

- Assignment operators are used to combine the '=' operator with one of the binary arithmetic operators.
- Assume c=9;

Operator	Example	Equivalent Statement	Results
+=	c += 7	c = c + 7	c = 16
-=	c -= 8	c = c –8	c = 1
*=	c *= 10	c = c * 10	c = 90
/=	c /= 5	c = c / 5	c = 1
%=	c %= 5	c = c % 5	c = 4

PROGRAM 3-4 Demonstration of Compound Assignments

```
1  /* Demonstrate examples of compound assignments.
2        Written by:
3        Date:
4  */
5  #include <stdio.h>
6
7  int main (void)
8  {
9  // Local Declarations
10    int x;
11    int y;
12
13 // Statements
14    x = 10;
15    y = 5;
16
```

PROGRAM 3-4 Demonstration of Compound Assignments

```
17      printf("x: %2d  |  y: %2d ",     x, y);
18      printf("  |  x *= y + 2: %2d ", x *= y + 2);
19      printf("  |  x is now: %2d\n",  x);
20
21      x = 10;
22      printf("x: %2d  |  y: %2d ",     x, y);
23      printf("  |  x /= y + 1: %2d ", x /= y + 1);
24      printf("  |  x is now: %2d\n",  x);
25
26      x = 10;
27      printf("x: %2d  |  y: %2d ",      x, y);
28      printf("  |  x %%= y - 3: %2d ", x %= y - 3);
29      printf("  |  x is now: %2d\n", x);
30
31      return 0;
32  }   // main
```

PROGRAM 3-4 Demonstration of Compound Assignments

```
Results:
x: 10  |  y:  5  |  x *= y + 2: 70  |  x is now: 70
x: 10  |  y:  5  |  x /= y + 1:  1  |  x is now:  1
x: 10  |  y:  5  |  x %= y - 3:  0  |  x is now:  0
```

Classification of Operator Continued…

Increment /Decrement Operator

- The Unary operator ++, --, is used as increment and decrement which acts upon single operand.

- Increment operator increases the value of variable by one.

- Similarly decrement operator decrease the value of the variable by one.

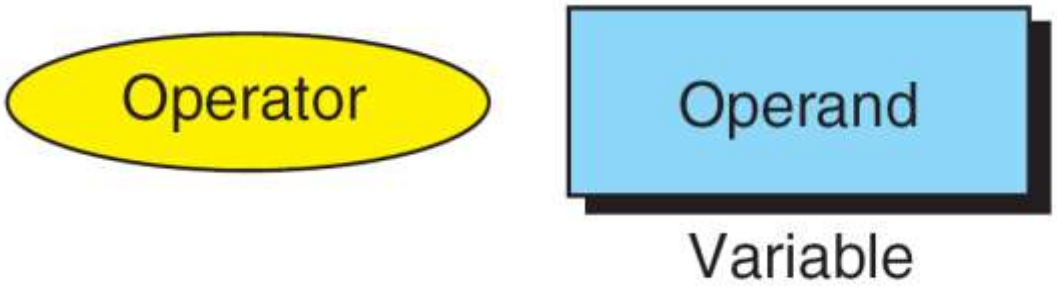
- And these operator can only used with the variable, but can't use with expression and constant as ++6 or ++(x+y+z).

Classification of Operator Continued…

Increment /Decrement Operator

- It again categories into Prefix Postfix.

Prefix Expression :

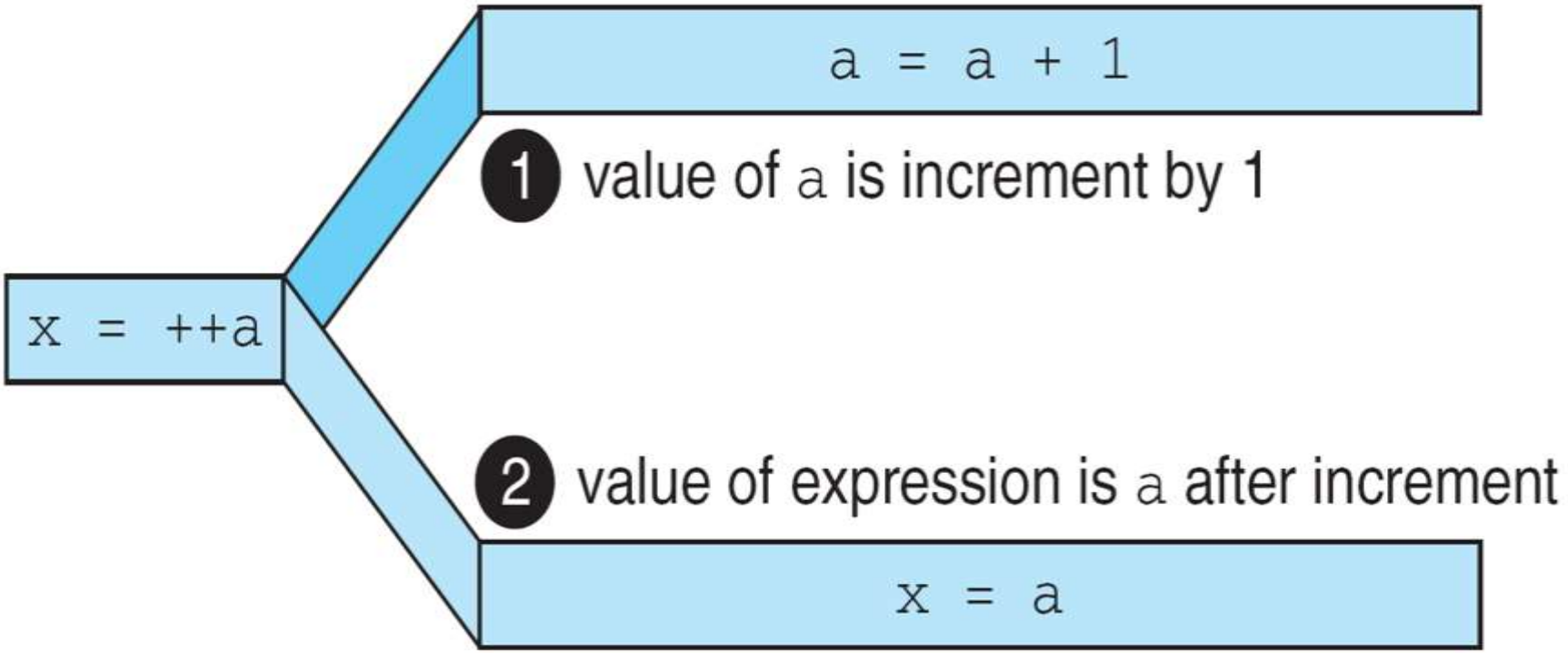


Classification of Operator Continued…

Note

The operand of a prefix expression must be a variable.

Classification of Operator Continued...



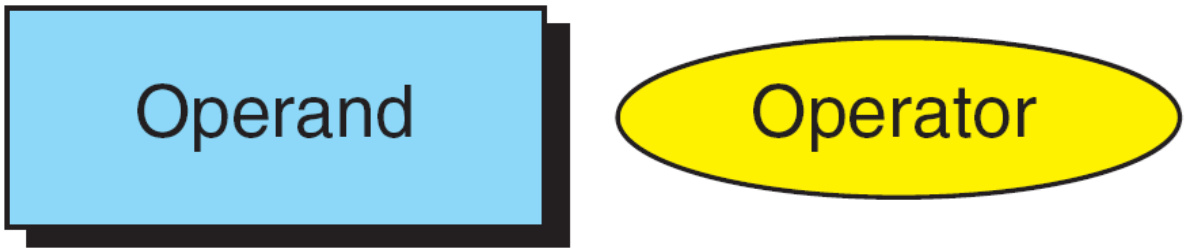
Result of Prefix ++a

Classification of Operator Continued…

Increment /Decrement Operator

- It again categories into Prefix Postfix.

Postfix Expression :



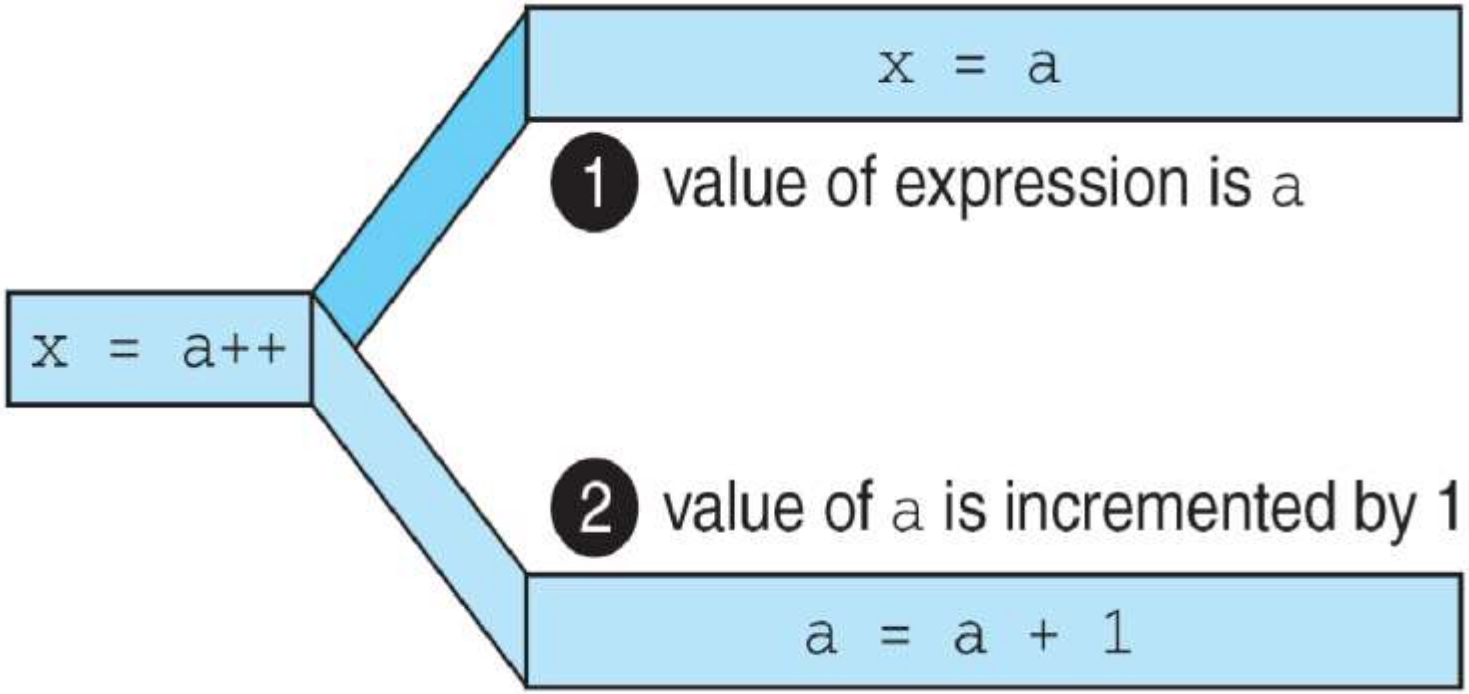
Variable

Classification of Operator Continued...

Note

(a++) has the same effect as (a = a + 1)

Classification of Operator Continued...



Result of Postfix a++

Classification of Operator Continued…

Increment /Decrement Operator

Symbol	Operation	Usage
++	postincrement	x++
--	postdecrement	x--
++	preincrement	++x
--	predecrement	--x

- **Pre:** Increment/decrement variable before using its value.
- **Post:** Increment/decrement variable after using its value.

Classification of Operator Continued…

Increment /Decrement Operator

- Example :

```
x = 4;
y = x++;
```

Results: x = 5, y = 4 (because x is incremented after assignment)

```
x = 4;
y = ++x;
```

Results: x = 5, y = 5 (because x is incremented before assignment)

Classification of Operator Continued…

Note

If ++ is after the operand, as in a++, the increment takes place after the expression is evaluated.

If ++ is before the operand, as in ++a, the increment takes place before the expression is evaluated.

Classification of Operator Continued…

Bitwise Operator

- Bitwise operator permit programmer to access and manipulate of data at bit level.

Symbol	Operation	Usage
~	bitwise NOT	~x
<<	left shift	x << y
>>	right shift	x >> y
&	bitwise AND	x & y
^	bitwise XOR	x ^ y
|	bitwise OR	x | y

Classification of Operator Continued…

Bitwise Operator

First Bit	Second Bit	AND (&)	OR (|)	XOR (^)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Bitwise AND : 3 = 00000011 , 5 = 00000101

3 & 5 = 00000011

<u>00000101</u>

00000001 -> 1

Classification of Operator Continued…

Bitwise Operator

Bitwise OR : 3 = 00000011 , 5 = 00000101

3 | 5 = 00000011

<u>00000101</u>

00000111 -> 7

Bitwise NOT : 3 = 00000011

~ 3 = 11111100

The bitwise not operator (~) invert zeros to ones and ones to zeros which in effect creates a negative value in binary.

Classification of Operator Continued…

Bitwise Operator

Bitwise shift operators are also supported

- << shift left and fill with zeros
- >> shift right with sign (copy 0 or 1)
- >>> shift right and fill with zero.

Classification of Operator Continued…

```
#include <stdio.h>

main() {

  unsigned int a = 60;   /* 60 = 0011 1100 */
  unsigned int b = 13;   /* 13 = 0000 1101 */
  int c = 0;

  c = a & b;       /* 12 = 0000 1100 */
  printf("Line 1 - Value of c is %d\n", c );

  c = a | b;       /* 61 = 0011 1101 */
  printf("Line 2 - Value of c is %d\n", c );

  c = a ^ b;       /* 49 = 0011 0001 */
  printf("Line 3 - Value of c is %d\n", c );

  c = ~a;        /*-61 = 1100 0011 */
  printf("Line 4 - Value of c is %d\n", c );

  c = a << 2;     /* 240 = 1111 0000 */
  printf("Line 5 - Value of c is %d\n", c );

  c = a >> 2;     /* 15 = 0000 1111 */
  printf("Line 6 - Value of c is %d\n", c );
}
```

Output:

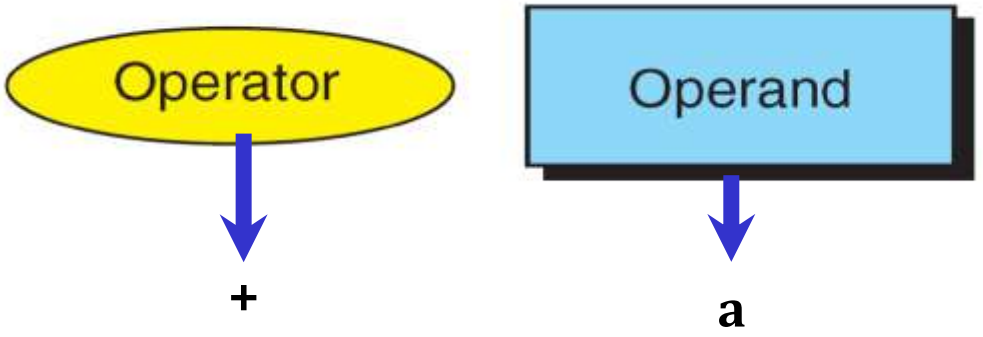
Line 1 - Value of c is 12
Line 2 - Value of c is 61
Line 3 - Value of c is 49
Line 4 - Value of c is -61
Line 5 - Value of c is 240
Line 6 - Value of c is 15

Classification of Operator Continued…

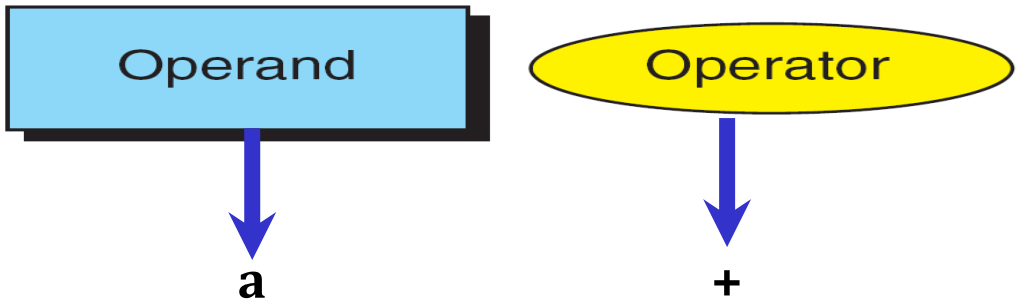
Unary Operator

- Some Operator required **single operand** to perform an operation called Unary Operator.
- Unary operators can use either **prefix** or **postfix** notation.

Prefix Unary Expression



Postfix Unary Expression



Classification of Operator Continued…

Unary Operator

Operator	Use	Description
+	+op	Positive
-	-op	Negative
++	++op	increments op by 1; evaluates to value of op before the incrementation
++	op++	increments op by 1; evaluates to value of op after the incrementation

Classification of Operator Continued…

Unary Operator

Operator	Use	Description
--	--op	decrements op by 1; evaluates to value of op before the decrementation
--	op--	decrements op by 1; evaluates to value of op after the decrementation

Classification of Operator Continued…

Unary Operator

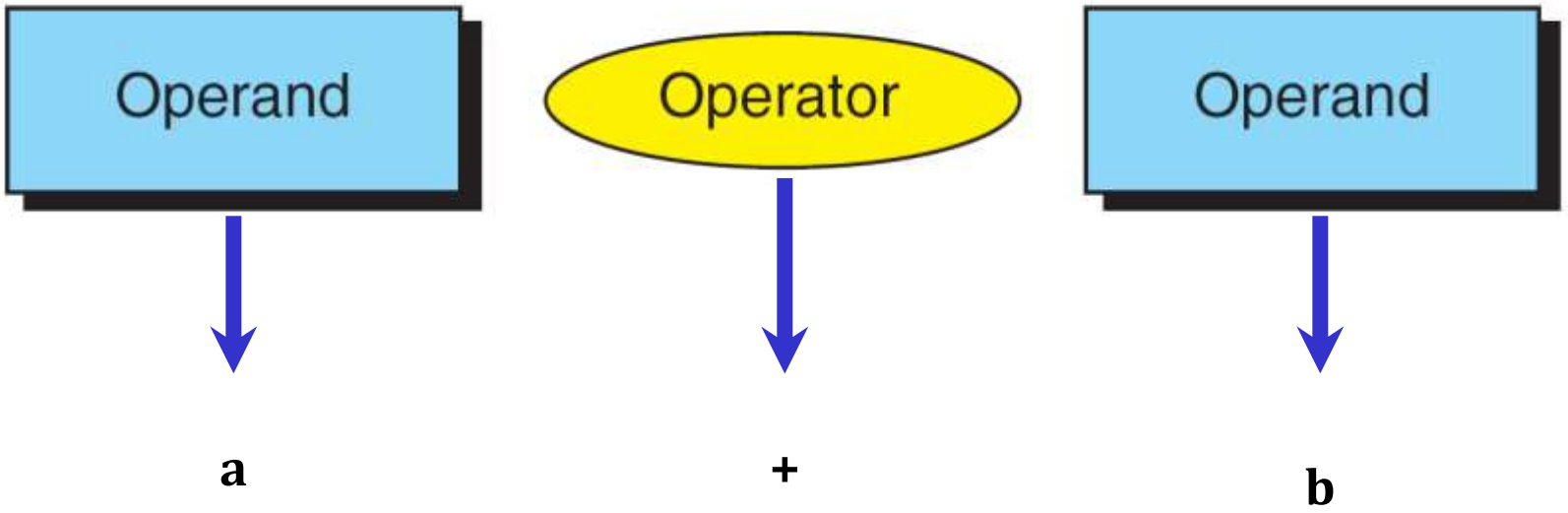
Expression	Contents of a Before *and* After Expression	Expression Value
+a	3	+3
−a	3	−3
+a	−5	−5
−a	−5	+5

Examples of Unary Plus And Minus Expressions

Classification of Operator Continued…

Binary Operator

- Some operator required two operand to perform an operation called Binary Operator.
- Example : Arithmetic Operator, Relational Operator, Logical Operator.



PROGRAM 3-3 Binary Expressions

```
1  /* This program demonstrates binary expressions.
2        Written by:
3        Date:
4  */
5  #include <stdio.h>
6  int main (void)
7  {
8  // Local Declarations
9     int   a = 17;
10    int   b = 5;
11    float x = 17.67;
12    float y = 5.1;
13
14 // Statements
15    printf("Integral calculations\n");
16    printf("%d + %d  = %d\n", a, b, a + b);
```

PROGRAM 3-3 Binary Expressions (continued)

```
17      printf("%d - %d  = %d\n", a, b, a - b);
18      printf("%d * %d  = %d\n", a, b, a * b);
19      printf("%d / %d  = %d\n", a, b, a / b);
20      printf("%d %% %d  = %d\n", a, b, a % b);
21      printf("\n");
25      printf("%f - %f  = %f\n", x, y, x - y);
26      printf("%f * %f  = %f\n", x, y, x * y);
27      printf("%f / %f  = %f\n", x, y, x / y);
28      return 0;
29  }   // main
```

PROGRAM 3-3 Binary Expressions (continued)

```
Results:
Integral calculations
17 + 5 = 22
17 - 5 = 12
17 * 5 = 85
17 / 5 = 3
17 % 5 = 2

Floating-point calculations
17.670000 + 5.100000  = 22.770000
17.670000 - 5.100000  = 12.570000
17.670000 * 5.100000  = 90.116997
17.670000 / 5.100000  = 3.464706
```

Classification of Operator Continued…

Note

Both operands of the modulo operator (%) must be integral types.

Classification of Operator Continued…

Ternary Operator (?:)

- The conditional operator (?:) is used to simplify an if/else statement.
- Syntax :

  Condition ? True Expression1 : False Expression2

- Example :

  (n > 1) ? (a + b) : (a * b)

  When (n>1) then the result is (a+b), otherwise the result is (a*b).

Classification of Operator Continued…

Short-hand Operator

- A **Shorthand operator** is a **shorter way** to express something that is **already available** in the **C programming language.**

- Short-hand Operator : +=,-=,*= and /=.

Compound Expression	Equivalent Simple Expression
x *= expression	x = x * expression
x /= expression	x = x / expression
x %= expression	x = x % expression
x += expression	x = x + expression
x -= expression	x = x - expression

Classification of Operator Continued…

Note

The left operand in an assignment expression must be a single variable.

Type Conversion

Type Conversion

- Converting an expression of a given type into another type is known as *type-casting*.
- Type conversion occurs when the expression has data of **mixed data types**.
- Example of such expression include converting an integer value in to a float value, or assigning the value of the expression to a variable with different data type.
- In type conversion, the data type is promoted from lower to higher because converting higher to lower involves loss of precision and value.

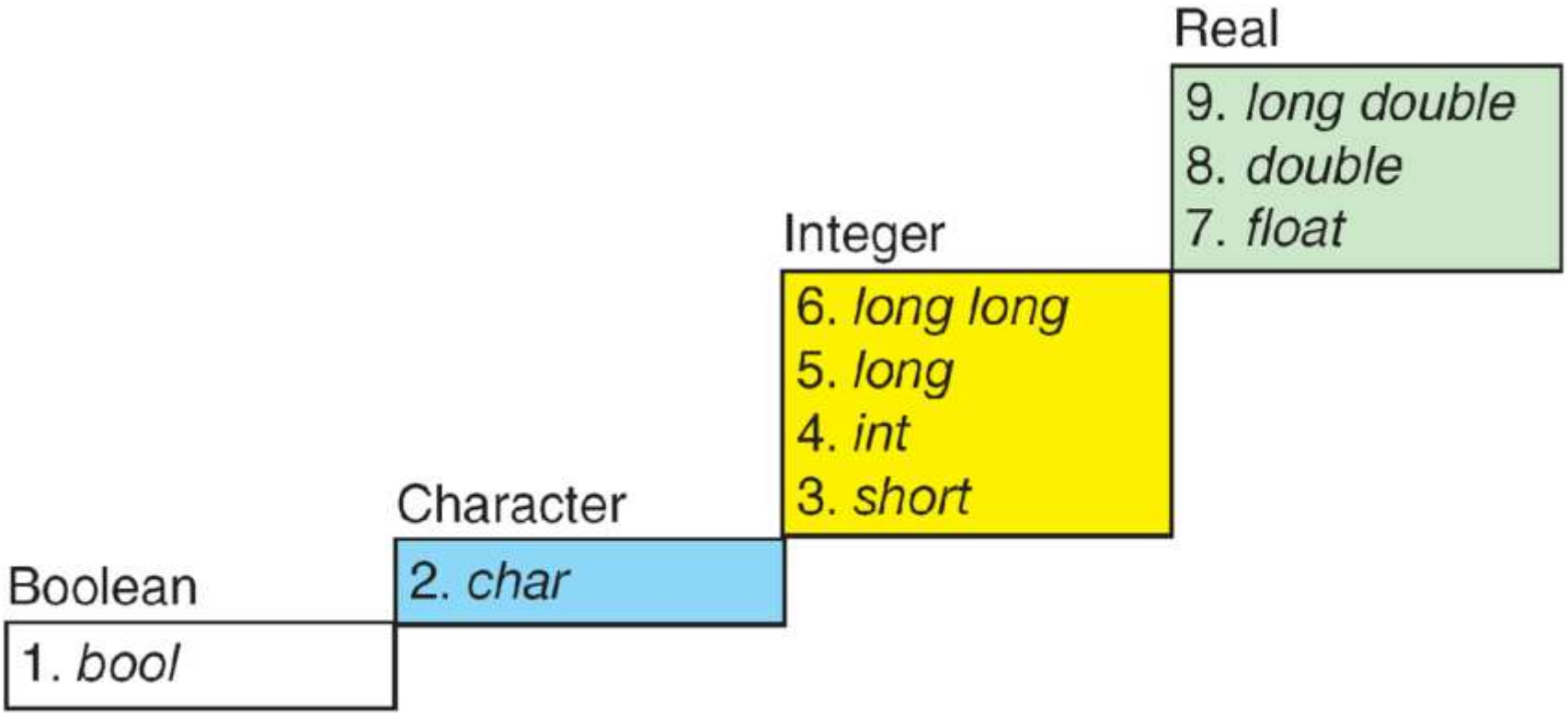
Type Conversion

Type Conversion

- For type conversion, C following some **General rules** explained below

1. Integer types are lower than floating point types
2. Signed types are lower than unsigned types
3. Short whole number types are lower than longer types
4. **double>float>long>int>short>char**

Type Conversion



Conversion Rank

Type Conversion Continued…

Type Conversion

- While Programming consider the following points

1. An arithmetic operation between an integer and integer always yields an integer result.

2. An operation between a float and float always yields a float result

3. An operation between an integer and float always yields a float result. In this operation the integer is first promoted to a float and then the operation is performed and the final result is a float.

int/int=int

float/int=float

int/float=float

float/float=float

Type Conversion Continued...

4. If the expression contains one operand as double data type and another operand as some other lower data type then the other operand is also converted to double and the result will be double. double operator (float(or)long(or)int(or)short)=>double

5. If the expression contains long and unsigned integer data types, the unsigned integer is converted to unsigned long and the result will be unsigned long.

6. Character and short data are promoted to integer

7. Unsigned char and unsigned short are converted to unsigned integer.

Type Conversion Continued…

Forced Conversion

- Forced conversion occurs when we are converting the value of the larger data type to the value of the smaller data type or smaller data type to the larger data type.
- For example, consider the following assignment statement ,

```
int a=5.5;
float b=100;
```

- In the first statement a=5.5; a is declared as int so the float value 5.5 cannot be stored in a. In such a case float is demoted to an int and then its value is stored. hence **5** is stored in a.
- In the second statement b=100; since b is a float variable 100 is promoted to **100.000000** and then stored in b.
- In general, the value of the expression is promoted or demoted depending on the type of variable on left hand side of =.

Type Conversion Continued...

Type Conversion

- There are two type of conversion :
  1. Implicit Type Conversion
  2. Explicit Type Conversion

1. **Implicit Type Conversion** : Implicit conversions do not require any operator. They are automatically performed when a value is copied to a compatible type.

```
float a=5.5;
int b;
b=a; // b will 5 after this
```

Type Conversion Continued…

2. **Explicit Type Conversion** : Explicit type conversions can be forced in any expression , with a unary operator called a cast.

- Syntax is : **(type-name) expression;**

- Example :

```
int n = 8;
float x;
x=(float)n; // x will 8.000000 after this
```

- The above statement will convert the value of n to a float value before assigning to x. but n is not altered.

PROGRAM 3-7 Implicit Type Conversion

```
1  /* Demonstrate automatic promotion of numeric types.
2        Written by:
3        Date:
4  */
5  #include <stdio.h>
6  #include <stdbool.h>
7
8  int main (void)
9  {
10 // Local Declarations
11    bool  b  = true;
12    char  c  = 'A';
13    float d  = 245.3;
14    int   i  = 3650;
15    short s  = 78;
16
```

PROGRAM 3-7 Implicit Type Conversion

```
17  // Statements
18     printf("bool + char is char:    %c\n", b + c);
19     printf("int * short is int:     %d\n", i * s);
20     printf("float * char is float: %f\n", d * c);
21
22     c = c + b;                      // bool promoted to char
23     d = d + c;                      // char promoted to float
24     b = false;
25     b = -d;                         // float demoted to bool
26
27     printf("\nAfter execution...\n");
28     printf("char + true:     %c\n", c);
29     printf("float + char:    %f\n", d);
30     printf("bool = -float:   %d\n", b);
31
32     return 0;
33  }  // main
```

PROGRAM 3-7 Implicit Type Conversion

```
Results:
bool + char is char:    B
int * short is int:     284700
float * char is float: 15944.500000

After execution...
char + true:     B
float + char:    311.299988
bool = -float:   1
```

Implicit Conversion of Ints and Floats

```
int i = 3;
float f = 2.5;

i = f;       // i will equal 2 after this

f =  3/i;   // f will equal 1 after this

f = i + 0.6; // f will equal 2.6 after this

i = 2.0 * 2.4; // i will equal 4 after this
```

PROGRAM 3-8 Explicit Casts

```
 1  /* Demonstrate casting of numeric types.
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6
 7  int main (void)
 8  {
 9  // Local Declarations
10     char   aChar    = '\0';
11     int    intNum1  = 100;
12     int    intNum2  =  45;
13     double fltNum1  = 100.0;
14     double fltNum2  =  45.0;
15     double fltNum3;
16
17  // Statements
18     printf("aChar numeric   :  %3d\n",   aChar);
```

PROGRAM 3-8 Explicit Casts

```
19      printf("intNum1 contains:  %3d\n",   intNum1);
20      printf("intNum2 contains:  %3d\n",   intNum2);
21      printf("fltNum1 contains:  %6.2f\n", fltNum1);
22      printf("fltNum2 contains:  %6.2f\n", fltNum2);
23
24      fltNum3 = (double)(intNum1 / intNum2);
25      printf
26         ("\n(double)(intNum1 / intNum2): %6.2f\n",
27          fltNum3);
28
29      fltNum3 = (double)intNum1 / intNum2;
30      printf("(double) intNum1 / intNum2 : %6.2f\n",
31             fltNum3);
32
33      aChar = (char)(fltNum1 / fltNum2);
34      printf("  (char)(fltNum1 / fltNum2): %3c\n",   aChar);
35
```

PROGRAM 3-8 Explicit Casts

```
36      return 0;
37  }   // main
```

```
Results:
aChar numeric    :     0
intNum1 contains:   100
intNum2 contains:    45
fltNum1 contains:   100.00
fltNum2 contains:    45.00

(double)(intNum1 / intNum2):    2.00
(double) intNum1 / intNum2 :    2.22
  (char)(fltNum1 / fltNum2):    ☺
```

Arithmetic Expression & Evaluation

Arithmetic Expression :

- Arithmetic expressions consist of operators, operands, parentheses, and function calls

Evaluation of Expressions

- All expressions in parenthesis must be evaluated separately, and inside out.

- The operator **precedence** rules for operators in same sub expression: Unary + and - are evaluated first and *, /, % evaluated next

Arithmetic Expression & Evaluation

Evaluation of Expressions

- **Associativity** rule Unary operators in same sub expressions and at same precedence level (such as +, - or *, /) are evaluated right to left.

- Binary operators in same sub expressions and at same precedence level are evaluated left to right.

Precedence & Associativity

Precedence of operators

- Operator precedence determines which operator is performed first in an expression with more than one operators with different precedence.

- In C, precedence of arithmetic operators(*, %, /, +, -) is higher than relational operators(==, !=, >, <, >=, <=) and precedence of relational operator is higher than logical operators(&&, || and !).

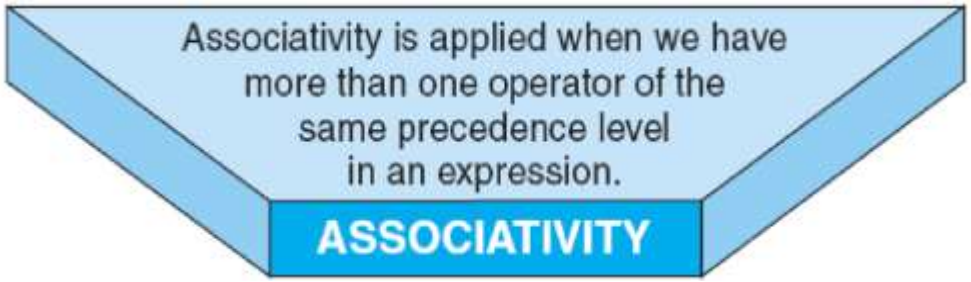
- For example **10 + 20 * 30 is calculated as 10 + (20 * 30)** and **not as (10 + 20) * 30.**

PROGRAM 3-5 Precedence

```
1  /* Examine the effect of precedence on an expression.
2        Written by:
```

PROGRAM 3-5 Precedence

```
Results:
a *  b + c  is: 230
a * (b + c) is: 500
```


Associativity is applied when we have more than one operator of the same precedence level in an expression.
ASSOCIATIVITY

Precedence & Associativity Continued…

Associativity of operators

- If two operators of same precedence (priority) is present in an expression, Associativity of operators indicate the order in which they execute.

- For Example : 1 == 2 != 3

- Here, operators == and != have same precedence. The associativity of both == and != is left to right, i.e, the expression on the left is executed first and moves towards the right.

- ((1 == 2) != 3)
  i.e, (1 == 2) executes first resulting into 0 (false) then,
  (0 != 3) executes resulting into 1 (true) ⍰ Output : 1

Precedence & Associativity Continued…

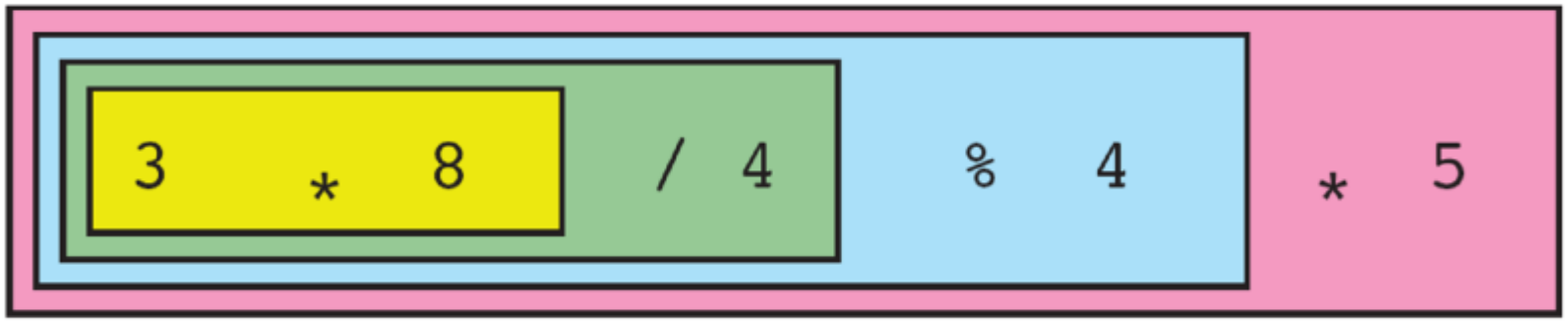
Highest

Lowest

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type) * & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Precedence & Associativity Continued…

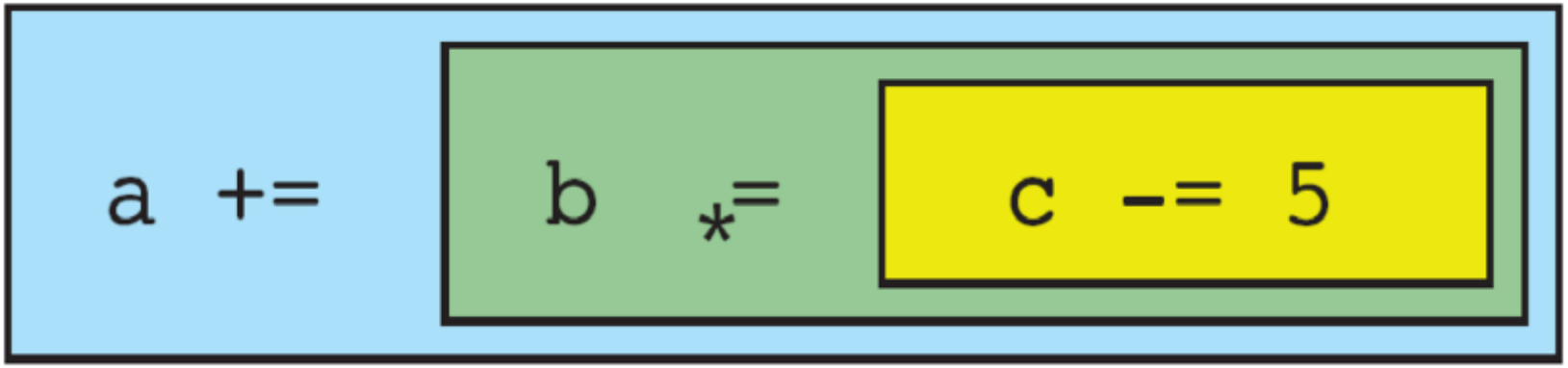
Associativity of operators



Left-to-Right Associativity

Precedence & Associativity Continued…

Associativity of operators



Right-to-Left Associativity

PROGRAM 3-6 Evaluating Expressions

```
1  /* Evaluate two complex expressions.
2        Written by:
3        Date:
4  */
5  #include <stdio.h>
6  int main (void)
7  {
8  // Local Declarations
9     int a = 3;
10    int b = 4;
11    int c = 5;
12    int x;
13    int y;
14
15 // Statements
16    printf("Initial values of the variables: \n");
17    printf("a = %d\tb = %d\tc = %d\n\n", a, b, c);
18
```

PROGRAM 3-6 Evaluating Expressions

```
19      x = a * 4 + b / 2 - c * b;
20      printf
21        ("Value of  a *  4 + b  / 2 - c  * b:  %d\n", x);
22
23      y = --a * (3 + b) / 2 - c++ * b;
24      printf
25        ("Value of --a * (3 + b) / 2 - c++ * b: %d\n", y);
26      printf("\nValues of the variables are now: \n");
27      printf("a = %d\tb = %d\tc = %d\n\n", a, b, c);
28
29      return 0;
30  }   // main
```

PROGRAM 3-6 Evaluating Expressions

```
Results:
Initial values of the variables:
a = 3    b = 4    c = 5

Value of    a *  4 + b  / 2 - c    * b:  -6
Value of --a * (3 + b) / 2 - c++ * b: -13

Values of the variables are now:
a = 2    b = 4    c = 6
```

a * 4 + b / 2 - c * b
3 * 4 + 4 / 2 – 5 * 4
12 + 4 / 2 – 5 * 4
12 + 2 – 5 * 4
12 + 2 – 20
14 - 20
-6

--a * (3 + b) / 2 - c++ * b
--a * 7 / 2 – c++ * b
2 * 7 / 2 - 5 * b
2 * 7 / 2 - 5 * b
2 * 7 / 2 - 5 * b
14 / 2 - 5 * 4
7 - 5 * 4
7 - 20
-13

Comma operator

- It can be used to link the related expressions together.
- A comma-linked list of expressions are evaluated *left to right* and the value of right most expression is the value of the combined expression.
- Example:

  value = (x=10,y=5,x+y);

  First assigns the value of 10 to x, then assigns 5 to y and finally assigns 15 to value

  Comma operator has the lowest precedence of all the operators, the parentheses are necessary.

The sizeof operator

- It is a compile time operator and, when used with an operand, it returns the number of bytes the operand occupies.
- The operand may be a variable, a constant or a data type qualifier.
- Examples:

```
m=sizeof(sum);
n=sizeof(long int);
```

3-7 Sample Programs

This section contains several programs that you should study for programming technique and style.

PROGRAM 3-9 Calculate Quotient and Remainder

```
1  /* Calculate and print quotient and remainder of two
2     numbers.
3        Written by:
4        Date:
5  */
6  #include <stdio.h>
7
8  int main (void)
9  {
10 // Local Declarations
11    int  intNum1;
12    int  intNum2;
13    int  intCalc;
14
15 // Statements
16    printf("Enter two integral numbers: ");
17    scanf ("%d %d", &intNum1, &intNum2);
18
```

PROGRAM 3-9 Calculate Quotient and Remainder

```
19      intCalc = intNum1 / intNum2;
20      printf("%d / %d is %d", intNum1, intNum2, intCalc);
21
22      intCalc = intNum1 % intNum2;
23      printf(" with a remainder of: %d\n", intCalc);
24
25      return 0;
26  }   // main
```

```
Results:
Enter two integral numbers: 13 2
13 / 2 is 6 with a remainder of: 1
```

PROGRAM 3-10 Print Right Digit of Integer

```
1  /* Print rightmost digit of an integer.
2        Written by:
3        Date:
4  */
5  #include <stdio.h>
6
7  int main (void)
8  {
9  // Local Declarations
10    int  intNum;
11    int  oneDigit;
12
13 // Statements
14    printf("Enter an integral number: ");
15    scanf ("%d", &intNum);
16
17    oneDigit = intNum % 10;
18    printf("\nThe right digit is: %d", oneDigit);
```

PROGRAM 3-10 Print Right Digit of Integer

```
19
20      return 0;
21  }  // main
```

```
Results:

Enter an integral number: 185

The right digit is: 5
```

PROGRAM 3-11 Calculate Average of Four Numbers

```
 1  /* Calculate the average of four integers and print
 2     the numbers and their deviation from the average.
 3        Written by:
 4        Date:
 5  */
 6  #include <stdio.h>
 7  int main (void)
 8  {
 9  // Local Declarations
10     int     num1;
11     int     num2;
12     int     num3;
13     int     num4;
14     int     sum;
15     float   average;
16
```

PROGRAM 3-11 Calculate Average of Four Numbers

```
17  // Statements
18      printf("\nEnter the first number  : ");
19      scanf("%d", &num1);
20      printf("Enter the second number : ");
21      scanf("%d", &num2);
22      printf("Enter the third  number : ");
23      scanf("%d", &num3);
24      printf("Enter the fourth number : ");
25      scanf("%d", &num4);
26
27      sum     = num1 + num2 + num3 + num4;
28      average = sum / 4.0;
29
30      printf("\n ******** average is %6.2f ******** ",
31              average);
32      printf("\n");
```

PROGRAM 3-11 Calculate Average of Four Numbers

```
33
34      printf("\nfirst number:  %6d -- deviation: %8.2f",
35              num1, num1 - average);
36      printf("\nsecond number: %6d -- deviation: %8.2f",
37              num2, num2 - average);
38      printf("\nthird number:  %6d -- deviation: %8.2f",
39              num3, num3 - average);
40      printf("\nfourth number: %6d -- deviation: %8.2f",
41              num4, num4 - average);
42
43      return 0;
44  }   // main
```

PROGRAM 3-11 Calculate Average of Four Numbers

```
Results:
Enter the first number:  23
Enter the second number: 12
Enter the third  number: 45
Enter the fourth number: 23

******** average is 25.75  ********

first number:      23 -- deviation:    -2.75
second number:     12 -- deviation:   -13.75
third number:      45 -- deviation:    19.25
fourth number:     23 -- deviation:    -2.75
```

PROGRAM 3-13 Calculate Sales Total

```
1  /* Calculates the total sale given the unit price,
2     quantity, discount, and tax rate.
3        Written by:
4        Date:
5  */
6  #include <stdio.h>
7
8  #define TAX_RATE 8.50
9
10 int main (void)
11 {
12 // Local Declarations
13    int    quantity;
14
15    float  discountRate;
16    float  discountAm;
17    float  unitPrice;
18    float  subTotal;
19    float  subTaxable;
```

PROGRAM 3-13 Calculate Sales Total

```
20      float  taxAm;
21      float  total;
22
23  // Statements
24      printf("\nEnter number of items sold:          ");
25      scanf("%d", &quantity);
26
27      printf("Enter the unit price:                 ");
28      scanf("%f", &unitPrice);
29
30      printf("Enter the discount rate (per cent): ");
31      scanf("%f", &discountRate);
32
33      subTotal    = quantity * unitPrice;
34      discountAm  = subTotal * discountRate / 100.0;
35      subTaxable  = subTotal - discountAm;
36      taxAm = subTaxable * TAX_RATE/ 100.00;
37      total = subTaxable + taxAm;
38
39      printf("\nQuantity sold:       %6d\n", quantity);
```

PROGRAM 3-13 Calculate Sales Total

```
40      printf("Unit Price of items: %9.2f\n", unitPrice);
41      printf("                     ------------\n");
42
43      printf("Subtotal :            %9.2f\n", subTotal);
44      printf("Discount:            -%9.2f\n", discountAm);
45      printf("Discounted total:     %9.2f\n", subTaxable);
46      printf("Sales tax:           +%9.2f\n", taxAm);
47      printf("Total sale:           %9.2f\n", total);
48
49      return 0;
50  }   // main
```

PROGRAM 3-13 Calculate Sales Total

```
Results:
Enter number of items sold:          34
Enter the unit price:                12.89
Enter the discount rate (per cent): 7

Quantity sold:            34
Unit Price of items:      12.89
                      ------------
Subtotal :              438.26
Discount:           -    30.68
Discounted total:       407.58
Sales tax:          +    34.64
Total sale:             442.23
```

PROGRAM 3-14 Calculate Student Score

```
1  /* Calculate a student's average score for a course
2     with 4 quizzes, 2 midterms, and a final. The quizzes
3     are weighted 30%, the midterms 40%, & the final 30%.
4        Written by:
5        Date:
6  */
7  #include <stdio.h>
8
9  #define QUIZ_WEIGHT      30
10 #define MIDTERM_WEIGHT   40
11 #define FINAL_WEIGHT     30
12 #define QUIZ_MAX        400.00
13 #define MIDTERM_MAX     200.00
14 #define FINAL_MAX       100.00
15
16 int main (void)
17 {
```

PROGRAM 3-14 Calculate Student Score

```
18  // Local Declarations
19      int     quiz1;
20      int     quiz2;
21      int     quiz3;
22      int     quiz4;
23      int     totalQuiz;
24      int     midterm1;
25      int     midterm2;
26      int     totalMidterm;
27      int     final;
28
29      float   quizPercent;
30      float   midtermPercent;
31      float   finalPercent;
32      float   totalPercent;
33
```

PROGRAM 3-14 Calculate Student Score

```
34  // Statements
35     printf("=========== QUIZZES ==================\n");
36     printf("Enter the score for the first quiz:  ");
37     scanf("%d", &quiz1);
38     printf("Enter the score for the second quiz: ");
39     scanf("%d", &quiz2);
40     printf("Enter the score for the third quiz:  ");
41     scanf("%d", &quiz3);
42     printf("Enter the score for the fourth quiz: ");
43     scanf("%d", &quiz4);
44
45     printf("============= MIDTERM  =============\n");
46     printf("Enter the score for the first midterm:  ");
47     scanf("%d", &midterm1);
48     printf("Enter the score for the second midterm: ");
49     scanf("%d", &midterm2);
50
```

PROGRAM 3-14 Calculate Student Score

```
51      printf("================ FINAL ==============\n");
52      printf("Enter the score for the final: ");
53      scanf("%d", &final);
54      printf("\n");
55
56      totalQuiz = quiz1 + quiz2 + quiz3 + quiz4;
57      totalMidterm = midterm1 + midterm2;
58
59      quizPercent =
60         (float)totalQuiz * QUIZ_WEIGHT / QUIZ_MAX;
61      midtermPercent =
62         (float)totalMidterm * MIDTERM_WEIGHT / MIDTERM_MAX;
63      finalPercent =
64         (float)final * FINAL_WEIGHT / FINAL_MAX;
65
66      totalPercent =
67            quizPercent + midtermPercent + finalPercent;
68
```

PROGRAM 3-14 Calculate Student Score

```
69      printf("First Quiz  %4d\n",   quiz1);
70      printf("Second Quiz %4d\n",   quiz2);
71      printf("Third Quiz  %4d\n",   quiz3);
72      printf("Fourth Quiz %4d\n",   quiz4);
73      printf("Quiz Total  %4d\n\n", totalQuiz);
74
75      printf("First Midterm  %4d\n",   midterm1);
76      printf("Second Midterm %4d\n",   midterm2);
77      printf("Total Midterms %4d\n\n", totalMidterm);
78
79      printf("Final          %4d\n\n", final);
80
81      printf("Quiz    %6.1f%%\n" , quizPercent);
82      printf("Midterm %6.1f%%\n" , midtermPercent);
83      printf("Final   %6.1f%%\n" , finalPercent);
84      printf("---------------\n");
85      printf("Total   %6.1f%%\n" , totalPercent);
86
87      return 0;
88  }   // main
```

PROGRAM 3-14 Calculate Student Score

```
Results:
========== QUIZZES ================
Enter the score for the first quiz:  98

Enter the score for the second quiz: 89
Enter the score for the third quiz:  78
Enter the score for the fourth quiz: 79
============ MIDTERM  ============
Enter the score for the first midterm:  90
Enter the score for the second midterm: 100
============== FINAL =============
Enter the score for the final: 92
```

PROGRAM 3-14 Calculate Student Score

```
First Quiz     98
Second Quiz    89
Third Quiz     78
Fourth Quiz    79
Quiz Total    344

First Midterm     90
Second Midterm   100
Total Midterms   190

Final             92

Quiz        25.8%
Midterm     38.0%
Final       27.6%
--------------
Total       91.4%
```

Previous year Questions

- **Fill in the blanks**

1. _____ finds the remainder of an integer division.
2. ____ operator reverses the value of the expression.
3. sizeof is a _____ operator used to calculate the size of data types.
4. _____ is also known as forced conversion
5. The sign of the result is positive in modulo division if _____
6. Associativity of operator defines _____
7. _____ is used to change the order of evaluating the expressions.
8. _____ operator returns the number of bytes occupied by the operand.
9. The operator which compares two values is _____
10. Ternary operator operates on _____ number of operands
11. _____ produces the 1s complement of the given binary value.
12. _____ operator has the lowest precedence
13. _____ operator cannot be used with float operands

Chapter – 3 : Operators and Expression in 'C'

Previous year Questions

- **State True or False**

1. The equality operators have higher precedence than the relational operators.
2. Shifting once to the left multiplies the number by 2.
3. All arithmetic operators have the same precedence.
4. The modulus operator can be used only with integers.

- **Review Questions**

1. Write a short note on operators available in C language.
2. Give the operator precedence chart.
3. Differentiate between type casting and type conversion.